

# A Lanczos Method for Approximating Composite Functions

Paul G. Constantine<sup>b,\*</sup>, Eric T. Phipps<sup>a</sup>

<sup>a</sup>*Sandia National Laboratories, Albuquerque, NM 87185*

<sup>b</sup>*Stanford University, Stanford, CA 94305*

---

## Abstract

We seek to approximate a composite function  $h(\mathbf{x}) = g(f(\mathbf{x}))$  with a global polynomial. The standard approach chooses points  $\mathbf{x}$  in the domain of  $f$  and computes  $h(\mathbf{x})$  at each point, which requires an evaluation of  $f$  and an evaluation of  $g$ . We present a Lanczos-based procedure that implicitly approximates  $g$  with a polynomial of  $f$ . By constructing a quadrature rule for the density function of  $f$ , we can approximate  $h(\mathbf{x})$  using many fewer evaluations of  $g$ . The savings is particularly dramatic when  $g$  is much more expensive than  $f$  or the dimension of  $x$  is large. We demonstrate this procedure with two numerical examples: (i) an exponential function composed with a rational function and (ii) a Navier-Stokes model of fluid flow with a scalar input parameter that depends on multiple physical quantities.

*Keywords:* dimension reduction, Lanczos' method, orthogonal polynomials, Gaussian quadrature

---

## 1. Introduction & Motivation

Many complex multiphysics models employ composite functions, where each member function represents a different physical model. For example,

---

\*Building 500, Room 501-L, Tel: (650) 723-2330

*Email addresses:* paul.constantine@stanford.edu (Paul G. Constantine), ethipp@sandia.gov (Eric T. Phipps)

<sup>1</sup>Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

consider a simple chemical reaction model; the decay of the concentration depends on the decay rate parameter, but the model for the decay rate (i.e., the Arrhenius model) depends on the temperature, the gas constant, the activation energy, and the prefactor. The concentration is then a function of the inputs to the rate model. Studying the effects of these inputs on the concentration could be challenging due to the large number of combinations of the four rate model inputs. However, one may make use of the composite structure – namely, that the concentration depends on these four inputs only through the rate model – to find a set of values for the rate parameter with which to study the effects on the concentration. By taking advantage of this structure, one can hope to reduce the cost of such a study by using fewer evaluations of the models.

We consider the general setting of a composite function

$$h(\mathbf{x}) = g(f(\mathbf{x})), \quad \mathbf{x} = (x_1, \dots, x_d) \in \mathcal{X} \subset \mathbb{R}^d \quad (1)$$

where

$$f : \mathcal{X} \longrightarrow \mathcal{F} \subset \mathbb{R} \quad (2)$$

$$g : \mathcal{F} \longrightarrow \mathcal{G} \subset \mathbb{R}. \quad (3)$$

One may be interested in understanding how  $h$  behaves as  $\mathbf{x}$  changes. However, if evaluating  $h$  is computationally expensive, then studies that require many evaluations may be infeasible. A common approach to this situation is to construct a surrogate approximation of  $h$  whose evaluations are cheaper given inputs  $\mathbf{x}$ . With a fixed number of  $h$  evaluations, one determines the parameters (e.g., coefficients) of the surrogate model, and then the surrogate is used to study the behavior of  $h$  with respect to changes in  $\mathbf{x}$ .

Global polynomials [1, 2] in  $\mathbf{x}$  are popular models for surrogate functions due to their rapid convergence rate as more terms are added to the approximation. For smooth functions, this often translates to relatively few evaluations to construct an accurate approximation. The approximation properties of univariate polynomials ( $d = 1$ ) are well-studied. Multivariate ( $d > 1$ ) polynomial approximations are less well understood, but a simple tensor product construction extends the univariate approximation characteristics to the multivariate setting. Unfortunately, such tensor product approximations suffer from the so-called curse of dimensionality. Loosely speaking, the number of function evaluations required to construct an accurate approximation increases exponentially as the dimension of  $\mathbf{x}$  increases. When evaluations of

$h$  are expensive, this often precludes the use of a multivariate polynomial surrogate.

In this work, we propose a strategy that takes advantage of the composite structure of  $h$  to reduce the overall cost of building a multivariate polynomial surrogate. Suppose one needs  $m$  points in  $\mathcal{X}$  to construct a multivariate polynomial surrogate of  $h(\mathbf{x})$ , where  $m$  may be an exponential function of  $d$ . Then each evaluation of  $h(\mathbf{x})$  requires first an evaluation of  $f(\mathbf{x})$  followed by an evaluation of  $g(f)$ , i.e.,  $2m$  total function evaluations.

Our proposed strategy computes the same  $m$  evaluations of  $f$ , but uses them to implicitly approximate a univariate density function on the range space  $\mathcal{F}$ . We then find  $k \ll m$  points in  $\mathcal{F}$  on which to evaluate  $g$ . The process of finding the  $k$  points in  $\mathcal{F}$  yields a transformation from the  $k$  evaluations of  $g$  to  $m$  approximations of  $h$ ; these  $m$  approximations of  $h$  are then used to compute the coefficients of a multivariate polynomial surrogate of  $h$ . We therefore reduce the cost of the construction from  $m$  evaluations of  $f$  and  $m$  evaluations of  $g$  to  $m$  evaluations of  $f$  and  $k \ll m$  evaluations of  $g$ , plus the cost of finding the  $k$  points and computing/applying the transformation. This is particularly advantageous when  $g(f)$  is much more expensive to evaluate than  $f(\mathbf{x})$ .

The  $k$  points that we find in  $\mathcal{F}$  are the Gaussian quadrature points from the implicitly approximated density function of  $f$ . The process constructs a set of polynomials in  $f$  that are orthonormal with respect to the density function of  $f$ . The function  $g$  is then approximated as a truncated series in these basis polynomials of  $f$ .

We use a discrete Stieltjes procedure [3] to compute the recurrence coefficients of the orthogonal polynomials in  $f$ , and we show how this is equivalent to a Lanczos' method [4, 5] on a diagonal matrix of the evaluations of  $f$  with a properly chosen starting vector. The basis vectors from the Lanczos iteration are used to linearly map the  $k$  evaluations of  $g(f)$  to  $m$  approximations of  $g(f(\mathbf{x}))$ , which are then used to approximate the coefficients of the multivariate polynomial surrogate of  $h$ .

A similar measure transformation approach was used in [6, 7] to approximate functions with sharp gradients. However, the theoretical and computational advantages gained by exploiting the connection to Lanczos' method were not explored.

In what follows, we review the polynomial approximation, Gaussian quadrature, and Lanczos' method (Section 2); define the problem and derive the approximation method (Section 3); and demonstrate its applicability on two

numerical examples: (i) an exponential function composed with a rational function and (ii) a Navier-Stokes model of fluid flow with a scalar input parameter that depends on multiple physical quantities (Section 4).

## 2. Preliminaries

In this section, we briefly review Gaussian quadrature and polynomial approximation, as well as Stieltjes' and Lanczos' methods. This will also serve to set up notation; we will use notation very similar to Gautschi [3].

### 2.1. Orthogonal polynomials, Gaussian quadrature, and pseudospectral approximation

Let  $\mathcal{T} \subset \mathbb{R}$  be equipped with a positive, normalized weight function  $\omega : \mathcal{T} \rightarrow \mathbb{R}_+$  with finite moments; denote an element of  $\mathcal{T}$  by  $t$ . For functions  $u(t)$  and  $v(t)$ , define the inner product

$$(u, v) = \int_{\mathcal{T}} u(t) v(t) \omega(t) dt \quad (4)$$

with associated norm  $\|u\| = (u, u)^{1/2}$ . Let  $\{\bar{\pi}_i(t)\}$  be the set of monic polynomials (i.e., leading coefficient is 1) that are orthogonal with respect to  $\omega(t)$ ,

$$(\bar{\pi}_i, \bar{\pi}_j) = \begin{cases} \|\bar{\pi}_i\|^2, & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

The monic orthogonal polynomials satisfy the recurrence relationship

$$\bar{\pi}_{i+1}(t) = (t - \alpha_i)\bar{\pi}_i(t) - \beta_i\bar{\pi}_{i-1}(t), \quad i = 0, 1, 2, \dots, \quad (6)$$

with  $\bar{\pi}_{-1}(t) = 0$  and  $\bar{\pi}_0(t) = 1$ . The  $\alpha_i$  and  $\beta_i$  are given by

$$\alpha_i = \frac{(t\bar{\pi}_i, \bar{\pi}_i)}{(\bar{\pi}_i, \bar{\pi}_i)}, \quad i = 0, 1, 2, \dots, \quad (7)$$

$$\beta_i = \frac{(\bar{\pi}_i, \bar{\pi}_i)}{(\bar{\pi}_{i-1}, \bar{\pi}_{i-1})}, \quad i = 1, 2, \dots \quad (8)$$

It is often more convenient to work with orthonormal instead of monic orthogonal polynomials, which we write as

$$\pi_i(t) = \frac{\bar{\pi}_i(t)}{\|\bar{\pi}_i\|}. \quad (9)$$

The recurrence relationship for the orthonormal polynomials becomes

$$\sqrt{\beta_{i+1}}\pi_{i+1}(t) = (t - \alpha_i)\pi_i(t) - \sqrt{\beta_i}\pi_{i-1}(t), \quad i = 0, 1, 2, \dots, \quad (10)$$

with  $\pi_{-1}(t) = 0$  and  $\pi_0(t) = 1$ . If we consider only the first  $n$  equations, then

$$t\pi_i(t) = \sqrt{\beta_i}\pi_{i-1}(t) + \alpha_i\pi_i(t) + \sqrt{\beta_{i+1}}\pi_{i+1}(t), \quad i = 0, 1, \dots, n-1. \quad (11)$$

Setting  $\boldsymbol{\pi}(t) = [\pi_0(t), \pi_1(t), \dots, \pi_{n-1}(t)]^T$ , we can write this conveniently in matrix form as

$$t\boldsymbol{\pi}(t) = \mathbf{J}\boldsymbol{\pi}(t) + \sqrt{\beta_n}\pi_n(t)\mathbf{e}_n, \quad (12)$$

where  $\mathbf{e}_n$  is a vector of zeros with a one in the last entry, and  $\mathbf{J} = \mathbf{J}_n$  (known as the *Jacobi matrix*) is an  $n \times n$  symmetric, tridiagonal matrix

$$\mathbf{J} = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_{n-2}} & \alpha_{n-2} & \sqrt{\beta_{n-1}} \\ & & & \sqrt{\beta_{n-1}} & \alpha_{n-1} \end{bmatrix}. \quad (13)$$

The zeros  $\{\lambda_j\}$  of  $\pi_n(t)$  are the eigenvalues of  $\mathbf{J}$  and  $\boldsymbol{\pi}(\lambda_j)$  are the corresponding eigenvectors; this follows directly from (12). Let  $\mathbf{Q}$  be the orthogonal matrix of eigenvectors of  $\mathbf{J}$ ; the elements of  $\mathbf{Q}$  are given by

$$\mathbf{Q}(i, j) = \frac{\pi_i(\lambda_j)}{\|\boldsymbol{\pi}(\lambda_j)\|_2}, \quad i, j = 0, \dots, n-1, \quad (14)$$

where  $\|\cdot\|_2$  is the standard 2-norm on  $\mathbb{R}^n$ . We write the eigenvalue decomposition of  $\mathbf{J}$  as

$$\mathbf{J} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T. \quad (15)$$

It is known that the eigenvalues  $\{\lambda_j\}$  are the nodes of the  $n$ -point Gaussian quadrature rule associated with the weight function  $\omega(t)$ . The quadrature weight  $\nu_j$  corresponding to  $\lambda_j$  is equal to the square of the first component of the eigenvector associated with  $\lambda_j$ ,

$$\nu_j = \mathbf{Q}(0, j)^2 = \frac{1}{\|\boldsymbol{\pi}(\lambda_j)\|_2^2}. \quad (16)$$

The weights  $\{\nu_j\}$  are known to be strictly positive. It will be notationally convenient to define the matrix  $\mathbf{W} = \text{diag}([\sqrt{\nu_0}, \dots, \sqrt{\nu_{n-1}}])$ .

For an integrable scalar function  $f(t)$ , we can approximate its integral by an  $n$ -point Gaussian quadrature rule. Let  $f_j = f(\lambda_j)$ . The quadrature approximation is a weighted sum of function evaluations,

$$\int_{\mathcal{T}} f(t) \omega(t) dt = \sum_{j=0}^{n-1} f_j \nu_j + R(f). \quad (17)$$

If  $f(t)$  is a polynomial of degree less than or equal to  $2n - 1$ , then  $R(f) = 0$ ; that is to say the *degree of exactness* of the Gaussian quadrature rule is  $2n - 1$ .

A square integrable function  $f(t)$  admits a mean-squared convergent Fourier series in the orthonormal polynomials. A pseudospectral approximation of  $f(t)$  is constructed by first truncating its Fourier series at  $n$  terms and approximating each Fourier coefficient with a quadrature rule. If we use the  $n$ -point Gaussian quadrature, then we can write

$$f(t) \approx \sum_{i=0}^{n-1} \hat{f}_i \pi_i(t), \quad (18)$$

where

$$\hat{f}_i = \sum_{j=0}^{n-1} f_j \pi_i(\lambda_j) \nu_j. \quad (19)$$

Let  $\mathbf{f} = [f_0, \dots, f_{n-1}]^T$  and  $\hat{\mathbf{f}} = [\hat{f}_0, \dots, \hat{f}_{n-1}]^T$ . Then using  $\mathbf{Q}$  from (15), (18), and (19), we can write

$$\hat{\mathbf{f}} = \mathbf{QWf} \quad f(t) \approx \hat{\mathbf{f}}^T \boldsymbol{\pi}(t) = \mathbf{f}^T \mathbf{W}^T \mathbf{Q}^T \boldsymbol{\pi}(t). \quad (20)$$

The expression  $\hat{\mathbf{f}} = \mathbf{QWf}$  is the discrete Fourier transform for the orthogonal polynomial basis. Note that it is easy to show that the pseudospectral approximation interpolates  $f(t)$  at the Gaussian quadrature points.

### 2.1.1. Tensor product extensions

The above concepts extend to multivariate functions via a tensor product construction. Let  $\mathcal{T} = \mathcal{T}_1 \otimes \dots \otimes \mathcal{T}_d$  be the domain with elements  $\mathbf{t} = (t_1, \dots, t_d)$ . We assume that the weight function  $\omega : \mathcal{T} \rightarrow \mathbb{R}_+$  is separable, i.e.  $\omega(\mathbf{t}) = \omega_1(t_1) \dots \omega_d(t_d)$ , where each univariate weight function is normalized and positive with finite moments.

Tensor product Gaussian quadrature rules are constructed by taking cross products of univariate Gaussian quadrature rules:

$$\boldsymbol{\lambda}_{i_1, \dots, i_d} = (\lambda_{i_1}, \dots, \lambda_{i_d}), \quad (21)$$

where the points  $\lambda_{i_r}$  with  $i_r = 0, \dots, n_r - 1$  are the univariate quadrature points for  $\omega_r(t_r)$ , with  $r = 1, \dots, d$ . The associated quadrature weights are given by the products

$$\nu_{i_1, \dots, i_d} = \nu_{i_1} \cdots \nu_{i_d}. \quad (22)$$

To approximate the integral of  $f(\mathbf{t})$ , compute

$$\int_{\mathcal{T}} f(\mathbf{t}) \omega(\mathbf{t}) d\mathbf{t} \approx \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} f(\boldsymbol{\lambda}_{i_1, \dots, i_d}) \nu_{i_1, \dots, i_d}. \quad (23)$$

The tensor product pseudospectral approximation is given by

$$f(\mathbf{t}) \approx \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} \hat{f}_{i_1, \dots, i_d} \pi_{i_1}(t_1) \cdots \pi_{i_d}(t_d), \quad (24)$$

where

$$\hat{f}_{i_1, \dots, i_d} = \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} f(\boldsymbol{\lambda}_{i_1, \dots, i_d}) \pi_{i_1}(\lambda_{j_1}) \cdots \pi_{i_d}(\lambda_{j_d}) \nu_{i_1, \dots, i_d}. \quad (25)$$

The matrix notation extends via Kronecker products. Let  $\boldsymbol{\pi}_{n_r}(t_r)$  be the vector of univariate polynomials that are orthonormal with respect to  $\omega_r(t_r)$  for  $r = 1, \dots, d$ . Then the vector

$$\boldsymbol{\pi}(\mathbf{t}) = \boldsymbol{\pi}_{n_1}(t_1) \otimes \cdots \otimes \boldsymbol{\pi}_{n_d}(t_d) \quad (26)$$

contains multivariate polynomials that are orthonormal with respect to  $\omega(\mathbf{t})$ . Similarly, define the matrices

$$\mathbf{Q} = \mathbf{Q}_{n_1} \otimes \cdots \otimes \mathbf{Q}_{n_d}, \quad \mathbf{W} = \mathbf{W}_{n_1} \otimes \cdots \otimes \mathbf{W}_{n_d}. \quad (27)$$

Then

$$\hat{\mathbf{f}} = \mathbf{Q} \mathbf{W} \mathbf{f} \quad f(\mathbf{t}) \approx \hat{\mathbf{f}}^T \boldsymbol{\pi}(\mathbf{t}) = \mathbf{f}^T \mathbf{W}^T \mathbf{Q}^T \boldsymbol{\pi}(\mathbf{t}), \quad (28)$$

where  $\hat{\mathbf{f}}$  is an  $(n_1 \cdots n_d)$ -vector of the tensor product pseudospectral coefficients,  $\mathbf{f}$  is an  $(n_1 \cdots n_d)$ -vector containing the evaluations of  $f$  at the points given by (21), ordered appropriately. Again, the expression  $\hat{\mathbf{f}} = \mathbf{Q} \mathbf{W} \mathbf{f}$  is the  $d$ -dimensional discrete Fourier transform for the polynomial basis.

## 2.2. Stieltjes' procedure

Stieltjes proposed a procedure for iteratively constructing a sequence of univariate polynomials that are orthogonal with respect to a given measure; see [3]. His method exploits the recurrence relationship for the orthogonal polynomials. He observed that with the weighted inner product (4), one may begin with  $\pi_{-1}$  and  $\pi_0$ , compute  $\alpha_0$  and  $\beta_1$  from (7) and (8), construct  $\pi_1$  from (6), compute  $\alpha_1$  and  $\beta_2$ , construct  $\pi_2$ , and so on.

A normalized version of Stieltjes' method for computing the orthonormal polynomials and their recurrence coefficients is given in Algorithm 1. The computed  $\alpha_i$  from Algorithm 1 are equivalent to the expression in (7), and the computed  $\eta_i$  are equal to  $\sqrt{\beta_i}$  in (8).

---

**Algorithm 1** A Stieltjes procedure for computing the first  $n$  orthonormal polynomials given a normalized weight function  $\omega(t)$ . Let  $\pi_{-1} = 0$  and  $\tilde{\pi}_0 = 1$ .

---

```

for  $i = 0$  to  $n - 1$  do
   $\eta_i = \|\tilde{\pi}_i\|$ 
   $\pi_i = \tilde{\pi}_i / \eta_i$ 
   $\alpha_i = (t\pi_i, \pi_i)$ 
   $\tilde{\pi}_{i+1} = (t - \alpha_i)\pi_i - \eta_i\pi_{i-1}$ 
end for

```

---

Gautschi [3] proposed to use a discrete inner product – e.g., based on a Gaussian quadrature rule. In the univariate case ( $d = 1$ ), this becomes

$$(u, v) \approx \sum_{j=0}^{m-1} u(\lambda_j) v(\lambda_j) \nu_j, \quad (29)$$

where  $\lambda_j$  and  $\nu_j$  are the points and weights of the discrete inner product. He reasoned that if the discrete inner product converges to the continuous, then the recurrence coefficients computed with the discrete inner product will also converge. Similarly, one may think of the recurrence coefficients computed with the discrete inner product as approximations of those computed with the continuous inner product.

In Section 3, we will use a tensor product quadrature rule to define a discrete inner product on the space  $\mathcal{X}$ . We will use this inner product in a Stieltjes procedure to compute the recurrence coefficients of a set of univariate



orthogonal polynomials, where orthogonality is with respect to the density function of  $f$  defined on  $\mathcal{F}$ . We have chosen to focus the presentation by using the tensor product quadrature rule to define the discrete inner product. However, the construction can be easily adjusted to employ other discrete inner products.

### 2.3. Lanczos method

Lanczos' method [4] for symmetric matrices is the foundation for iterative eigensolvers and Krylov subspace methods for solving symmetric linear systems. It generates a symmetric, tridiagonal matrix (the Jacobi matrix) and a sequence of mutually orthogonal (in exact arithmetic) vectors known as the Lanczos vectors. The eigenvalues of the tridiagonal matrix – known as the Ritz values – approximate the eigenvalues of the symmetric matrix.

In fact, Algorithm 1 is exactly a form of Lanczos' method<sup>2</sup>, if we replace (i) the variable  $t$  by a symmetric matrix  $\mathbf{A}$  of size  $m \times m$ , (ii) the polynomials  $\pi_i(t)$  by the Lanczos vectors  $\mathbf{v}_i$ , (iii) the starting polynomial  $\tilde{\pi}_0$  by a starting vector  $\tilde{\mathbf{v}}_0$ , and (iv) the inner product by a discrete, weighted inner product.

Suppose that  $k$  iterations of the method have been executed. We can write the recurrence relationship for the Lanczos vectors in matrix notation as

$$\mathbf{A}\mathbf{V} = \mathbf{V}\mathbf{T} + \eta_k \mathbf{v}_k \mathbf{e}_k^T, \quad (30)$$

where  $\mathbf{V} = [\mathbf{v}_0, \dots, \mathbf{v}_{k-1}]$  is an  $m \times k$  matrix of Lanczos vectors,  $\mathbf{T}$  is the  $k \times k$  symmetric, tridiagonal Jacobi matrix of recurrence coefficients, and  $\mathbf{e}_k$  is a last column of the  $k \times k$  identity matrix.

## 3. The approximation method

In this section, we take advantage of the relationship between an approximate Stieltjes' procedure with a discrete inner product and Lanczos' method to devise a computational method for approximating composite functions. Recall the problem setup from the introduction:

$$h(\mathbf{x}) = g(f(\mathbf{x})), \quad \mathbf{x} = (x_1, \dots, x_d) \in \mathcal{X} \subset \mathbb{R}^d \quad (31)$$

---

<sup>2</sup>However, Algorithm 1 has undesirable numerical properties as an implementation.

where

$$f : \mathcal{X} \longrightarrow \mathcal{F} \subset \mathbb{R} \quad (32)$$

$$g : \mathcal{F} \longrightarrow \mathcal{G} \subset \mathbb{R}. \quad (33)$$

We restrict our attention to input spaces that are  $d$ -dimensional hypercubes,  $\mathcal{X} = [-1, 1]^d$ , although this can be relaxed to more general hyperrectangles. We assume that  $\mathcal{X}$  is equipped with a positive, separable weight function  $\omega_{\mathbf{x}} = \omega(\mathbf{x}) = \omega_1(x_1) \cdots \omega_d(x_d)$  with finite moments. We also assume that  $f$  is bounded, so that  $\mathcal{F}$  is a closed interval in  $\mathbb{R}$ .

A tensor product pseudospectral approximation of  $h$  on the space  $\mathcal{X}$  follows the construction in Section 2.1.1:

$$h(\mathbf{x}) \approx \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} \hat{h}_{i_1, \dots, i_d} \pi_{i_1}(x_1) \cdots \pi_{i_d}(x_d), \quad (34)$$

where the  $\pi_{i_r}(x_r)$  are the univariate polynomials that are orthogonal with respect to  $\omega_r(x_r)$ . The coefficients  $\hat{h}_{i_1, \dots, i_d}$  are

$$\hat{h}_{i_1, \dots, i_d} = \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} h(\boldsymbol{\lambda}_{j_1, \dots, j_d}) \pi_{i_1}(\lambda_{j_1}) \cdots \pi_{i_d}(\lambda_{j_d}) \nu_{j_1, \dots, j_d}, \quad (35)$$

where  $\boldsymbol{\lambda}_{j_1, \dots, j_d} = (\lambda_{j_1}, \dots, \lambda_{j_d})$  and  $\nu_{j_1, \dots, j_d} = \nu_{j_1} \cdots \nu_{j_d}$  are the nodes and weights, respectively, of the tensor product Gaussian quadrature for the weight function  $\omega_{\mathbf{x}}$ . This can be written conveniently in matrix form as in (28) as

$$\hat{\mathbf{h}} = \mathbf{Q}_{\mathbf{x}} \mathbf{W}_{\mathbf{x}} \mathbf{h}, \quad h(\mathbf{x}) \approx \hat{\mathbf{h}}^T \boldsymbol{\pi}(\mathbf{x}) = \mathbf{h}^T \mathbf{W}_{\mathbf{x}}^T \mathbf{Q}_{\mathbf{x}}^T \boldsymbol{\pi}(\mathbf{x}), \quad (36)$$

where  $\hat{\mathbf{h}}$  is a vector of the tensor product pseudospectral coefficients;  $\mathbf{h}$  is a vector of evaluations of  $h$  at the quadrature points;  $\boldsymbol{\pi}(\mathbf{x})$  is a vector of the product type multivariate orthonormal polynomials; and  $\mathbf{W}_{\mathbf{x}}$  and  $\mathbf{Q}_{\mathbf{x}}$  represent the  $d$ -dimensional discrete Fourier transform for functions defined on  $\mathcal{X}$ .

We assume that the primary expense of this computation comes from the evaluations of  $h$  at the quadrature points. There are  $m = n_1 \cdots n_d$  points in the tensor product quadrature rule; if  $n_1 = \cdots = n_d = n$ , then  $m = n^d$ . Each evaluation of  $h$  requires an evaluation of  $f$  followed by an evaluation

of  $g$ . Our goal is to approximate the values of  $h$  at the quadrature points using  $m$  evaluations of  $f$ , but only  $k \ll m$  evaluations of  $g$ ; we accomplish this goal by taking advantage of the composite structure of  $h$ .

Our strategy is to apply Stieltjes' procedure to implicitly construct a set of polynomials that are orthonormal with respect to the density function of  $f$  defined on  $\mathcal{F}$ . The  $k$ -point Gaussian quadrature rule for the univariate density function provides a set of points in  $\mathcal{F}$  on which to evaluate  $g$ . We then linearly map the  $k$  evaluations of  $g$  to approximate the  $m$  evaluations of  $h$ .

Let  $\omega_f : \mathcal{F} \rightarrow \mathbb{R}_+$  be the normalized density function of  $f$ . We assumed  $f$  was bounded on  $\mathcal{X}$ , which implies  $\mathcal{F}$  is a closed interval and  $\omega_f$  is bounded. Therefore,  $\omega_f$  has finite moments, and it admits a set of univariate orthonormal polynomials  $\phi_i = \phi_i(f)$  with  $i = 0, 1, 2, \dots$ . Then we can approximate  $g = g(f)$  with a univariate pseudospectral approximation,

$$g(f) \approx \sum_{i=0}^{k-1} \hat{g}_i \phi_i(f), \quad (37)$$

where

$$\hat{g}_i = \sum_{j=0}^{k-1} g(\theta_j) \phi_i(\theta_j) \mu_j. \quad (38)$$

The  $\theta_j \in \mathcal{F}$  and  $\mu_j \in \mathbb{R}_+$  are the nodes and weights, respectively, of the  $k$ -point Gaussian quadrature rule for  $\omega_f$ . In the matrix notation,

$$\hat{\mathbf{g}} = \mathbf{Q}_f \mathbf{W}_f \mathbf{g}, \quad g(f) \approx \hat{\mathbf{g}}^T \boldsymbol{\phi}(f) = \mathbf{g}^T \mathbf{W}_f^T \mathbf{Q}_f^T \boldsymbol{\phi}(f), \quad (39)$$

where  $\hat{\mathbf{g}}$  is a vector of the pseudospectral coefficients;  $\mathbf{g}$  is a vector of evaluations of  $g$  at the quadrature points  $\theta_j$ ;  $\boldsymbol{\phi}(f)$  is a vector of the univariate orthonormal polynomials; and  $\mathbf{W}_f$  and  $\mathbf{Q}_f$  represent the discrete Fourier transform for functions defined on  $\mathcal{F}$ .

Notice that if we were given (i) the coefficients  $\hat{g}_i$  and (ii) the evaluations of the polynomials  $\phi_i$  at the point  $f(\mathbf{x})$  with  $\mathbf{x} \in \mathcal{X}$ , then

$$h(\mathbf{x}) = g(f(\mathbf{x})) \approx \sum_{i=0}^{k-1} \hat{g}_i \phi_i(f(\mathbf{x})). \quad (40)$$

Unfortunately, the direct evaluation of this expression is problematic, since each new point  $\mathbf{x}$  requires the computation of  $f(\mathbf{x})$ . We would prefer a

surrogate like (34) whose evaluation is independent of both  $g$  and  $f$ . With this in mind, we seek to evaluate (40) at only the quadrature points  $\boldsymbol{\lambda}_{i_1, \dots, i_d}$  on  $\mathcal{X}$ :

$$h(\boldsymbol{\lambda}_{j_1, \dots, j_d}) \approx \underline{h}(\boldsymbol{\lambda}_{j_1, \dots, j_d}) = \sum_{i=0}^{k-1} \hat{g}_i \phi_i(f(\boldsymbol{\lambda}_{j_1, \dots, j_d})). \quad (41)$$

We denote the output of the pseudospectral expansion (41) by  $\underline{h}$ . We write (41) in matrix notation as

$$\mathbf{h} \approx \underline{\mathbf{h}} = \mathbf{U} \hat{\mathbf{g}}. \quad (42)$$

The elements of the  $m \times k$  matrix  $\mathbf{U}$  are the evaluations of  $\phi_i(f(\boldsymbol{\lambda}_{i_1, \dots, i_d}))$ , where each row of  $\mathbf{U}$  corresponds to a quadrature point  $\boldsymbol{\lambda}_{i_1, \dots, i_d}$  and each column corresponds to a polynomial  $\phi_i(f)$ . This matrix enables a construction similar to (34).

Putting these ideas together, we construct a global polynomial surrogate for  $h(\mathbf{x})$  that approximates the tensor product pseudospectral approximation. The form of the polynomial surrogate is the same as the true pseudospectral approximation – i.e., a linear combination of the multivariate orthogonal basis polynomials – but the pseudospectral coefficients are approximated using  $k \ll m$  evaluations of  $g$  plus the cost of computing the polynomial evaluations  $\phi_i(f(\boldsymbol{\lambda}_{i_1, \dots, i_d}))$ . We will denote the approximate coefficients by  $\hat{\underline{h}} \approx \hat{h}$ , which is consistent with the notation  $\underline{h} \approx h$ .

More precisely, we construct a surrogate for  $h(\mathbf{x})$  as

$$h(\mathbf{x}) \approx \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} \hat{\underline{h}}_{i_1, \dots, i_d} \pi_{i_1}(x_1) \cdots \pi_{i_d}(x_d), \quad (43)$$

where

$$\hat{\underline{h}}_{i_1, \dots, i_d} = \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} \underline{h}(\boldsymbol{\lambda}_{j_1, \dots, j_d}) \pi_{i_1}(\lambda_{j_1}) \cdots \pi_{i_d}(\lambda_{j_d}) \nu_{j_1, \dots, j_d}. \quad (44)$$

The quantities  $\underline{h}(\boldsymbol{\lambda}_{j_1, \dots, j_d})$  are given by (41). In matrix notation, we combine (36), (39), and (42) to get

$$h(\mathbf{x}) \approx \hat{\underline{\mathbf{h}}}^T \boldsymbol{\pi}(\mathbf{x}) \quad (45)$$

$$= \underline{\mathbf{h}}^T \mathbf{W}_x^T \mathbf{Q}_x^T \boldsymbol{\pi}(\mathbf{x}) \quad (46)$$

$$= \hat{\mathbf{g}}^T \mathbf{U}^T \mathbf{W}_x^T \mathbf{Q}_x^T \boldsymbol{\pi}(\mathbf{x}) \quad (47)$$

$$= \mathbf{g}^T \mathbf{W}_f^T \mathbf{Q}_f^T \mathbf{U}^T \mathbf{W}_x^T \mathbf{Q}_x^T \boldsymbol{\pi}(\mathbf{x}), \quad (48)$$

where  $\hat{\mathbf{h}}$  is a vector of the coefficients  $\hat{h}_{i_1, \dots, i_d}$ , and  $\mathbf{h}$  is a vector of the evaluations of  $h$ . Notice that  $\mathbf{g}$  contains  $k$  evaluations of  $g$ . In what follows, we will show how to compute the points  $\theta_j$  in (38) and the transformations  $\mathbf{U}$ ,  $\mathbf{Q}_f$ , and  $\mathbf{W}_f$  from (48) using  $m$  evaluations  $f(\boldsymbol{\lambda}_{j_1, \dots, j_d})$ . Thus, we will approximate the tensor product pseudospectral approximation (34) using  $k$  evaluations of  $g$  and  $m$  evaluations of  $f$ .

### 3.1. Lanczos' method for approximation

The strategy is implemented computationally with Lanczos' method applied to a diagonal matrix whose nonzero elements are the  $m$  evaluations of  $f$  at the quadrature points on  $\mathcal{X}$ ; the  $k$ -point Gaussian quadrature rule on  $\mathcal{F}$  comes from the computed Jacobi matrix (see (15)), and the map from  $k$  evaluations of  $g$  to  $m$  evaluations of  $h$  comes from the Lanczos vectors.

It is notationally convenient in this section to order the  $d$ -dimensional quadrature points  $\boldsymbol{\lambda}_{i_1, \dots, i_d} \in \mathcal{X}$  so as to be indexed by the natural numbers. For the remainder, we will refer to a node as  $\boldsymbol{\lambda}_i$  with corresponding weight  $\nu_i$  for  $i = 0, \dots, m-1$ . The specific ordering must be consistent with the tensor product structure of (27). We will similarly order the function evaluations  $f_i = f(\boldsymbol{\lambda}_i)$ .

We first show that Lanczos' method yields the quantities desired from the Stieltjes procedure. The fact has been observed elsewhere in the literature [8, 9, 10, 11]; we state it as a theorem for reference and notation.

**Theorem 1.** *Let*

$$\mathbf{A} = \begin{bmatrix} f_0 & & \\ & \ddots & \\ & & f_{m-1} \end{bmatrix} \quad (49)$$

*be the diagonal matrix whose nonzero elements are the evaluations of  $f$  at the quadrature nodes  $\boldsymbol{\lambda}_i \in \mathcal{X}$ . For an  $m$ -vector of ones  $\mathbf{e}$  and a starting vector  $\tilde{\mathbf{v}}_0 = \mathbf{W}_{\mathbf{x}}\mathbf{e}$ , Lanczos method applied to  $\mathbf{A}$  is equivalent to Stieltjes' procedure with a discrete inner product to construct the recurrence coefficients of the polynomials  $\phi_i(f)$  that are orthonormal with respect to an approximation of the measure  $\omega_f$ . The discrete inner product is defined by the nodes  $\boldsymbol{\lambda}_i$  and weights  $\nu_i$  of the tensor product Gaussian quadrature rule for the measure  $\omega_{\mathbf{x}}$ .*

*Proof.* To prove this statement, we simply describe the quantities in Algorithm 1 with the discrete inner product. The starting polynomial  $\tilde{\phi}_0 = 1$

corresponds to an  $m$ -vector of ones,  $\mathbf{e}$ . Let  $\mathbf{v}_{-1}$  be an  $m$ -vector of zeros. Then the quantities from Algorithm 1 with the discrete inner product become

$$\begin{aligned}\eta_i &= \left( \sum_{j=0}^{m-1} \tilde{\phi}_i(f_j)^2 \nu_j \right)^{1/2} = (\tilde{\mathbf{v}}_i^T \tilde{\mathbf{v}}_i)^{1/2} \\ \sqrt{\nu_j} \phi_i(f_j) &= \frac{\tilde{\phi}_i(f_j)}{\eta_i}, \quad \mathbf{v}_i = [\sqrt{\nu_0} \phi_i(f_0), \dots, \sqrt{\nu_{m-1}} \phi_i(f_{m-1})]^T, \\ \alpha_i &= \sum_{j=0}^{m-1} f_j \phi_i(f_j)^2 \nu_j = \mathbf{v}_i^T \mathbf{A} \mathbf{v}_i, \\ \tilde{\phi}_{i+1}(f_j) &= (f_j - \alpha_i) \phi_i(f_j) - \eta_i \phi_{i-1}(f_j),\end{aligned}$$

which can be written

$$\tilde{\mathbf{v}}_{i+1} = (\mathbf{A} - \alpha_i \mathbf{I}) \mathbf{v}_i - \eta_i \mathbf{v}_{i-1}, \quad (50)$$

with  $\tilde{\mathbf{v}}_{i+1} = [\tilde{\phi}_{i+1}(f_0), \dots, \tilde{\phi}_{i+1}(f_{m-1})]^T$ . To recover the polynomials,

$$\mathbf{U} = \mathbf{W}_{\mathbf{x}}^{-1} \mathbf{V}, \quad (51)$$

where  $\mathbf{U}(j, i) = \phi_i(f_j)$ .  $\square$

The matrix  $\mathbf{U}$  from (51) is the same as the one from (42). Note that – as mentioned in Theorem 1 – these quantities do not correspond to the exact measure  $\omega_f$ . They instead correspond to an approximation of  $\omega_f$  from the evaluations  $f_j = f(\boldsymbol{\lambda}_j)$ . We will not examine the error made in this approximation; we assume the evaluations of  $f$  at the nodes  $\boldsymbol{\lambda}_j \in \mathcal{X}$  are sufficient to resolve the salient features of  $\omega_f$ .

Running  $k$  steps of the Lanczos process yields the recurrence relationship (30). The elements of the  $k \times k$  tridiagonal Jacobi matrix  $\mathbf{T}$  are the recurrence coefficients of the polynomials  $\phi_i(f)$  up to order  $k-1$ . The Lanczos vectors  $\mathbf{V}$  contain the polynomial evaluations  $\phi_i(f_j)$  scaled by  $\sqrt{\nu_j}$  as in (51).

Denote the eigendecomposition of  $\mathbf{T}$  by

$$\mathbf{T} = \mathbf{Q}_f \boldsymbol{\Theta} \mathbf{Q}_f^T, \quad \boldsymbol{\Theta} = \text{diag}([\theta_0, \dots, \theta_{k-1}]). \quad (52)$$

The Ritz values (the eigenvalues of  $\mathbf{T}$ ) are the Gaussian quadrature nodes for the approximation of  $\omega_f$  on the space  $\mathcal{F}$ , and the weights come from the

first component of the eigenvectors of  $\mathbf{T}$  as in (16). Precisely speaking, the  $\phi_i(f)$  are orthogonal with respect to the discrete measure defined by the  $\theta_j$  and  $\mu_j$ .

Therefore, we compute the quantities  $\theta_j$ ,  $\mu_j$ ,  $\mathbf{Q}_f$ ,  $\mathbf{W}_f$ , and  $\mathbf{U}$  from (48) using  $k$  steps of Lanczos' method applied to the diagonal matrix  $\mathbf{A}$  followed by the eigendecomposition of the Jacobi matrix  $\mathbf{T}$ . This is the computational cost incurred beyond the  $m$  evaluations of  $f$  and  $k$  evaluations of  $g$ .

### 3.2. Loss of orthogonality and stopping criteria

We have stated that we expect  $k \ll m$ , or that the number of points in the discrete measure on  $\mathcal{F}$  will be much smaller than the number of points in the discrete measure on  $\mathcal{X}$ . The number  $k$  is the number of iterations of the Lanczos procedure; how do we know how many iterations to use to get an accurate approximation of the measure on  $\mathcal{F}$ ?

It is well known that Lanczos' method in finite precision behaves differently than the algorithm in exact arithmetic; a thorough treatment of this subject can be found in Meurant's excellent monograph [5], as well as [12]. In particular, the Lanczos vectors will often lose orthogonality after some number of iterations.

Thanks to the work of Paige [13] as described in [5] – as well as [14, 15, 16] – we know that the loss of orthogonality is closely related to the convergence of the Ritz values to the true eigenvalues; loosely speaking, once a Ritz value has converged to an eigenvalue, the remaining Lanczos vectors lose orthogonality. It has been observed that in many cases the extremal Ritz values converge to the extremal eigenvalues fastest depending on the starting vector. From this we can expect that the Lanczos vectors will lose orthogonality once the extremal Ritz values are sufficiently close to the extremal eigenvalues. We use this expectation to motivate a heuristic for stopping the Lanczos iteration. Further justification of the following heuristic is the subject of on-going work.

Recall that  $\mathbf{A}$  is diagonal, so we are not concerned with any particular eigenvalue. In fact, we are only concerned with approximating the range of the data – which is the range of the function  $f(\mathbf{x})$  evaluated at the points  $\mathbf{\lambda}_i$  – and its corresponding measure. Therefore, once the extremal Ritz values converge, we are satisfied. Leveraging the work on Lanczos' method in finite precision, we can judge when the extremal Ritz values have converged by checking orthogonality of the Lanczos vectors. Essentially, we can treat the

loss of orthogonality in the Lanczos vectors as stopping criteria. We use the following measure of loss of orthogonality given a tolerance TOL:

$$\tau = \log_{10} (\|\mathbf{I} - \mathbf{V}^T \mathbf{V}\|_F) > \text{TOL} \quad (53)$$

where  $\|\cdot\|_F$  is the Frobenius norm. Other efficient measures for determining loss of orthogonality are discussed in [17, Chapter 9] as well as [18, 19]. In the numerical examples of Section 4, we choose TOL=-14.

If the iterations continue beyond this point, we find that the points and weights of the quadrature rule for the measure on  $\mathcal{F}$  become less smooth; this phenomenon is similar to choosing the wrong bin size for a histogram. In some cases, we observe the familiar (to those who have studied Lanczos' method) appearance of ghost eigenvalues. If we examine the weights corresponding to pairs of nearly identical Ritz values, we usually find that one of the weights is orders of magnitude smaller than the other. Of course, we would prefer to ignore points with very small weights, since this would correspond to a wasted function evaluation in the quadrature approximations. We demonstrate these phenomena on the following numerical examples.

### 3.3. An Algorithm

We close this section with a step-by-step algorithm using the linear algebra notation to summarize the procedure. Given functions  $f(\mathbf{x})$  and  $g(f)$ , the goal is to approximate the coefficients of a tensor product pseudospectral polynomial surrogate for the composite function  $h(\mathbf{x}) = g(f(\mathbf{x}))$ .

1. Obtain the  $m$  nodes  $\boldsymbol{\lambda}_i$  and weights  $\nu_i$  of the tensor product Gaussian quadrature rule for the space  $\mathcal{X}$ . Also, obtain the  $m \times m$  matrix  $\mathbf{Q}_{\mathbf{x}}$  and the diagonal matrix of the square root of the quadrature weights  $\mathbf{W}_{\mathbf{x}}$  from (28)<sup>3</sup>.
2. For  $i = 0, \dots, m-1$ , compute  $f_i = f(\boldsymbol{\lambda}_i)$ , and form the diagonal matrix  $\mathbf{A} = \text{diag}([f_0, \dots, f_{m-1}])$ .
3. For the starting vector  $\tilde{\mathbf{v}}_0 = \mathbf{W}_{\mathbf{x}} \mathbf{e}$ , run Lanczos' method until  $\tau > \text{TOL}$  from (53), and let  $k$  be the number of iterations. Store the matrix  $\mathbf{U}$  from (51).

---

<sup>3</sup>In a real implementation,  $\mathbf{Q}_{\mathbf{x}}$  and  $\mathbf{W}_{\mathbf{x}}$  do not need to be formed explicitly. We only need the action of  $\mathbf{Q}_{\mathbf{x}}$  on a vector, which can be computed efficiently using methods such as [20].



4. Compute the eigenvalue decomposition of the  $k \times k$  Jacobi matrix  $\mathbf{T}$  from the Lanczos procedure to get the quadrature nodes  $\{\theta_k\}$  and the discrete Fourier transform matrices  $\mathbf{Q}_f$  and  $\mathbf{W}_f$ ; see (52).
5. For  $i = 0, \dots, k-1$ , compute  $g_i = g(\theta_i)$  and form the vector  $\mathbf{g} = [g_0, \dots, g_{k-1}]^T$ .
6. Compute approximate coefficients for the pseudospectral approximation  $\hat{\mathbf{h}} \approx \hat{\mathbf{h}}$  as

$$\hat{\mathbf{h}} = \mathbf{Q}_x \mathbf{W}_x \mathbf{U} \mathbf{Q}_f \mathbf{W}_f \mathbf{g}. \quad (54)$$

These coefficients define a polynomial approximation of  $h(\mathbf{x})$  with a basis of multivariate product-type orthonormal polynomials.

#### 4. Numerical Examples

We present two numerical studies demonstrating the qualities of the method. The first is an example with functions chosen to stress the method's properties. The second applies the method – as a proof of concept – to a model from fluid dynamics with a scalar input parameter that depends on multiple physical quantities.

##### 4.1. Simple functions

Let  $\mathbf{x} = (x_1, x_2) \in [-1, 1]^2$  with a uniform measure of  $1/4$  in  $[-1, 1]^2$  and zero otherwise. Given parameters  $\delta_1 > 1$  and  $\delta_2 > 1$ , define the function

$$f(\mathbf{x}) = \frac{1}{(x_1 - \delta_1)(x_2 - \delta_2)}. \quad (55)$$

Notice that  $f(\mathbf{x}) > 0$ , and  $\delta_1$  and  $\delta_2$  determine how quickly  $f$  grows near the boundary. The closer  $\delta_1$  and  $\delta_2$  are to 1, the closer the singularity in the function gets to the domain, which determines how large  $f$  is at the point  $(x_1 = 1, x_2 = 1)$ . For the numerical experiments, we choose  $\delta_1 = \delta_2 = 1.3$ . The function  $f$  is analytic in  $\mathbf{x}$ , so we expect polynomial approximations to converge exponentially as the degree of approximation increases.

Next we choose  $g(f) = \exp(f)$ , so that

$$h(\mathbf{x}) = g(f(\mathbf{x})) = \exp\left(\frac{1}{(x_1 - \delta_1)(x_2 - \delta_2)}\right). \quad (56)$$

Again,  $g(f)$  is analytic in  $f$ , so  $h(\mathbf{x})$  is analytic in  $\mathbf{x}$ , as well.

We choose the discrete measure on  $\mathcal{X}$  to be a tensor product Gauss-Legendre quadrature rule on  $[-1, 1]^2$  with  $n$  points in each variable, which results in  $m = n^2$  points and weights. The  $m \times m$  diagonal matrix  $\mathbf{A}$  has diagonal elements equal to  $f$  evaluated at the points of the discrete measure.

To test the approximation properties of the Lanczos-enabled method, it is sufficient to examine the error at the tensor product quadrature points; see (41). This is essentially the same as computing the difference between the true pseudospectral coefficients  $\hat{h}$  and their approximation  $\hat{\underline{h}}$ . For the number  $m$  of bivariate nodes of the tensor product quadrature rule and the number  $k$  of Lanczos iterations, define the error  $\mathcal{E} = \mathcal{E}(m, k)$  as

$$\mathcal{E}^2 = \sum_{i=0}^{m-1} \left( h(\boldsymbol{\lambda}_i) - \sum_{j=0}^{k-1} \hat{g}_j \phi_j(f(\boldsymbol{\lambda}_i)) \right)^2 \quad (57)$$

$$= \|\mathbf{h} - \mathbf{U}\hat{\mathbf{g}}\|_2^2. \quad (58)$$

In Figure 1, we plot both  $\log_{10}(\mathcal{E})$  and the measure of orthogonality of the Lanczos vectors (see (53)) as  $m$  and  $k$  increase. To read these plots, choose  $m$  from the  $y$ -axis, and then follow the plot to the right to increase the Lanczos iteration  $k$ . It is interesting to note that the error in the approximation does not increase after the Lanczos vectors lose orthogonality. The loss of orthogonality is useful for a stopping criteria to determine the smallest  $k$  that produces an accurate approximation. However, taking more than the minimum number of Lanczos iterations does no harm for this example.

In Figure 2, we plot a series of bar graphs of the quadrature weights  $\mu_l$  at points  $\theta_l$  for the measure on  $\mathcal{F}$  computed with  $m = 81$ . While the bar plot resembles a histogram, the comparison between a histogram and quadrature weights is not precise. Nevertheless, the series of bar plots demonstrates the behavior of the weights as the Lanczos iteration index continues beyond the point when the Lanczos vectors lose orthogonality; the orthogonality measures from (53) are presented in each plot. We observe that the weights lose smoothness as the Lanczos vectors lose orthogonality; note the weights in the right tail of the plot.

#### 4.2. Fluid flow example

As an example of applying these techniques to an engineering problem, we examine a simple channel flow problem with a scalar input parameter (the Reynolds number) that depends on multiple physical quantities (density and

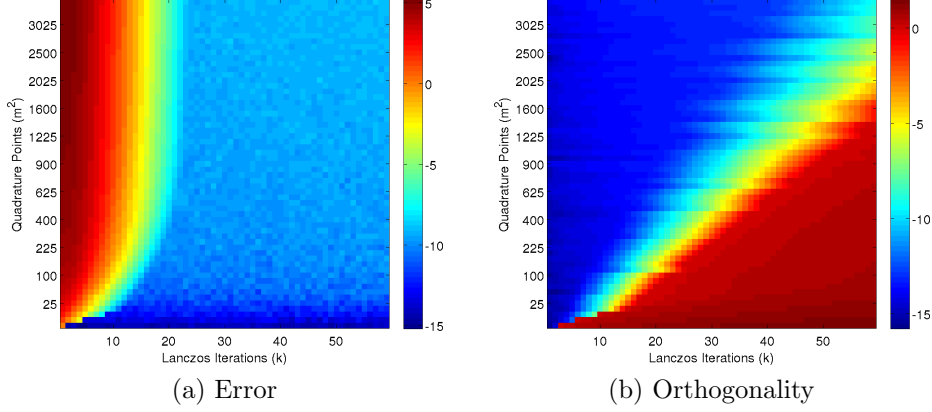


Figure 1: Figure 1a plots the error in approximation of  $h$  as measured by (57). Figure 1b shows the loss of orthogonality in the Lanczos vectors using  $\tau$  from (53).

viscosity). Consider the two-dimensional rectangular domain of length  $L = 1$  m and width  $W = 0.1$  m shown in Figure 3. Water flows into the left side of the domain with a horizontal velocity of  $u_0 = 0.01$  m/sec, and we are interested in computing the velocity of the flow out of the domain on the right side.

At room temperature and standard pressure, the dynamics of the fluid within the domain are well-modeled by the incompressible Navier-Stokes equations

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla \mathbf{u}) - \mu \nabla^2 \mathbf{u} + \nabla P = 0, \quad (59)$$

$$\rho(\nabla \cdot \mathbf{u}) = 0, \quad (60)$$

where  $\mathbf{u}$  is the two-component velocity of the fluid,  $\rho$  is the density,  $\mu$  is the viscosity, and  $P$  is the pressure. Using the inlet flow velocity and the width  $W$  of the domain, the equations are non-dimensionalized resulting in

$$\frac{\partial \bar{\mathbf{u}}}{\partial \bar{t}} + \bar{\mathbf{u}} \cdot \bar{\nabla} \bar{\mathbf{u}} - \frac{1}{Re} \bar{\nabla}^2 \bar{\mathbf{u}} + \bar{\nabla} \bar{P} = 0, \quad (61)$$

$$\bar{\nabla} \cdot \bar{\mathbf{u}} = 0, \quad (62)$$

where  $\bar{\nabla} = \nabla/W$ , and

$$Re = \frac{\rho u_0 W}{\mu} \quad (63)$$

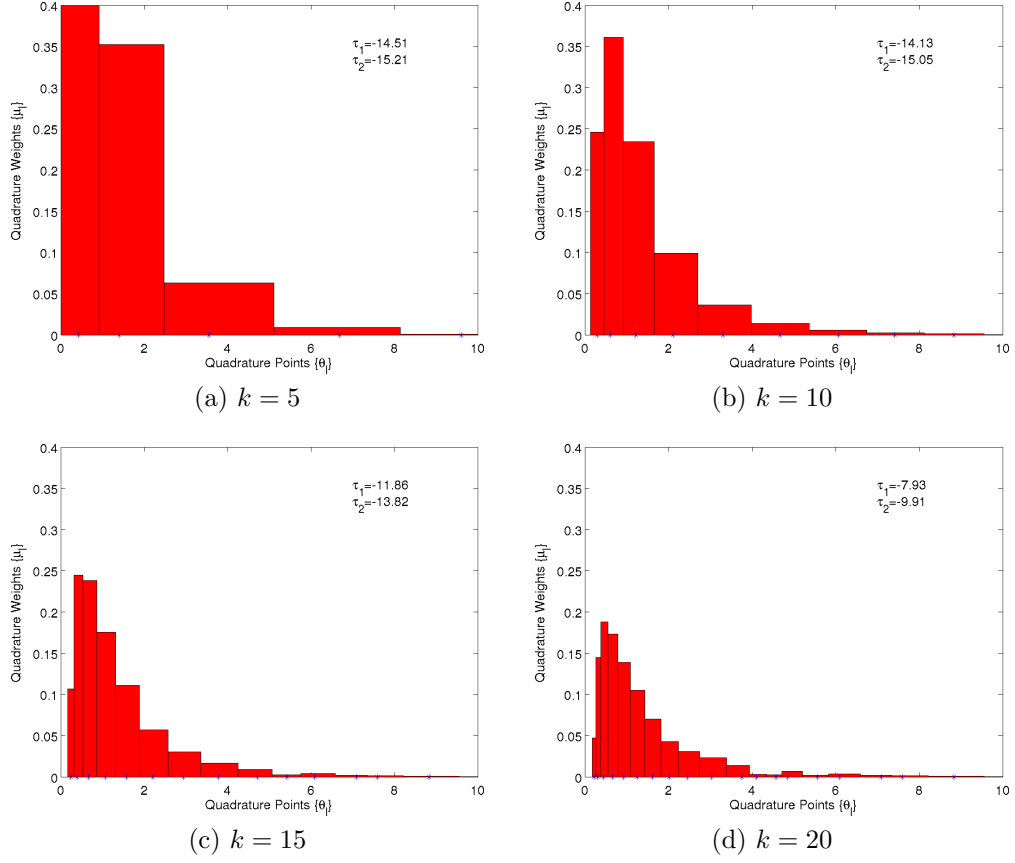


Figure 2: A series of bar plots showing the weights  $\mu_l$  at the points  $\theta_l$  for the quadrature rule on  $\mathcal{F}$ . The number  $\tau$  in each plot shows the measure of orthogonality in the Lanczos vectors; it is defined in (53).

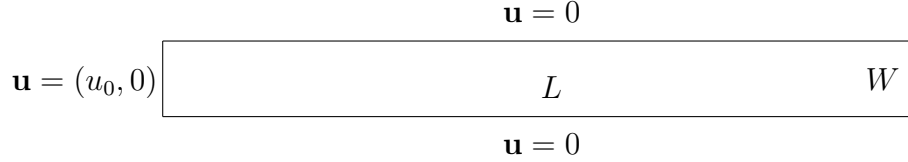


Figure 3: Fluid flow domain

is the Reynolds number.

Equations 61-62 are discretized spatially on a mesh of 500 by 50 quadrangle cells using the finite element method with piecewise bilinear basis functions for both the velocities and pressures [21]. Given a Reynolds number, the resulting nonlinear algebraic equations are solved via Newton's method using a GMRES linear solver [22] and incomplete-LU factorization preconditioner. The resulting flow solution at density  $\rho = \rho_0 = 998.205 \text{ kg/m}^3$  and viscosity  $\mu = \mu_0 = 0.001001 \text{ Ns/m}^2$  is shown in Figure 4; the density and viscosity values roughly correspond to water at room temperature and standard pressure. The calculations were implemented in the Albany [23] simulation package using numerous solver and discretization packages from the Trilinos framework [24].

We consider a problem where the ambient temperature and pressure are uncertain resulting in uncertain density and viscosity. In particular we model the density and viscosity as uniformly distributed random variables

$$\rho \in [0.99\rho_0, 1.01\rho_0] \quad \mu \in [0.9\mu_0, 1.1\mu_0]. \quad (64)$$

In other words, we assume density varies uniformly by 1% and viscosity varies uniformly by 10%. In the notation of Section 3, we have

$$\mathbf{x} = (\rho, \mu),$$

$$f(\mathbf{x}) = \frac{1}{Re} = \frac{\mu}{\rho u_0 W}.$$

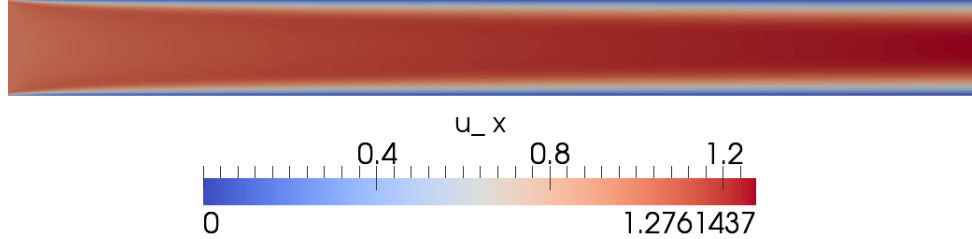


Figure 4: Horizontal flow velocity at mean density and viscosity

The function  $h(\mathbf{x}) = g(f(\mathbf{x}))$  corresponds to the maximum outflow velocity at the right side of the domain given fixed values for  $\rho$  and  $\mu$ . Each evaluation of  $g$  involves an expensive solution of equations 61-62 – compared to computing  $f(\mathbf{x})$ .

For this experiment, we choose a tensor product Gauss-Legendre quadrature rule with 11 points in the range of  $\rho$  and 11 points in the range of  $\mu$  for a total of  $m = 121$  points. We use the procedure from Section 3 to approximate the maximum outflow velocity at all 121 pairs of  $(\rho, \mu)$  by constructing a  $k$ -point Gaussian quadrature rule for  $1/Re$  with  $k = 13$ . In other words, with only 13 evaluations of  $g$  – the expensive flow solver – we can approximate the output  $h$  at 121 points in the parameter space corresponding to  $\mathbf{x}$ .

To check the error in the approximation, we also compute the maximum outflow velocity at all 121 combinations of  $\rho$  and  $\mu$ , which enables the computation of (57). With 13 steps of the Lanczos procedure, we have a loss of orthogonality in the basis vectors of  $\tau = -13.14$  (see equation (53)). The error in approximation (equation (57)) is  $1.55 \times 10^{-6}$ .

## 5. Conclusion

We have presented a method for approximating a composite function by implicitly approximating the outer function as a polynomial of the output of the inner function. This measure transformation is based on Stieltjes' method for generating orthogonal polynomials given an inner product, and

it is implemented as Lanczos' method on a diagonal matrix of inner function evaluations at the points of a discrete measure. We have developed a heuristic for when to terminate the Lanczos iteration based on the loss of orthogonality in the Lanczos vectors – a common phenomenon for the algorithm in finite precision. The resulting method reduces the number of evaluations of the outer function, which are only required at the Gaussian quadrature points of the transformed measure. The numerical experiments show the behavior of the method and the scale of the reduction.

## References

- [1] D. Xiu, G. E. Karniadakis, The Wiener-Askey polynomial chaos for stochastic differential equations, *SIAM Journal of Scientific Computing* 24 (2002) 619 – 644.
- [2] D. Xiu, J. S. Hesthaven, High order collocation methods for differential equations with random inputs, *SIAM Journal of Scientific Computing* 27 (2005) 1118 – 1139.
- [3] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, Clarendon Press, Oxford, 2004.
- [4] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *Journal of Research of the National Bureau of Standards* 45 (1950) 255 – 282.
- [5] G. Meurant, *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*, SIAM, 2006.
- [6] M. Gerritsma, J.-B. van der Steen, P. Vos, G. Karniadakis, Time-dependent generalized polynomial chaos, *Journal of Computational Physics* 229 (22) (2010) 8333 – 8363. doi:10.1016/j.jcp.2010.07.020.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999110004134>
- [7] G. Poette, D. Lucor, Non intrusive iterative stochastic spectral representation with application to compressible gas dynamics, *Journal of Computational Physics* (0) (2012) –. doi:10.1016/j.jcp.2011.12.038.  
URL <http://www.sciencedirect.com/science/article/pii/S0021999112000058>

- [8] W. B. Gragg, W. J. Harrod, The numerically stable reconstruction of jacobi matrices from spectral data, *Numerische Mathematik* 44 (1984) 317–335, 10.1007/BF01405565.  
URL <http://dx.doi.org/10.1007/BF01405565>
- [9] A. Greenbaum, *Iterative Methods for solving linear systems*, Vol. 17 of *Frontiers in Applied Mathematics*, SIAM, 1997.
- [10] D. OLeary, Z. Strako, P. Tich, On sensitivity of gausschristoffel quadrature, *Numerische Mathematik* 107 (2007) 147–174, 10.1007/s00211-007-0078-x.  
URL <http://dx.doi.org/10.1007/s00211-007-0078-x>
- [11] G. H. Golub, G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, 2010.
- [12] G. Meurant, Z. Strakoš, The Lanczos and conjugate gradient algorithms in finite precision arithmetic, *Acta Numerica* 15 (2006) 471 – 542.
- [13] C. C. Paige, The computation of eigenvalues and eigenvectors of very large sparse matrices, Ph.D. thesis, University of London (1971).
- [14] B. N. Parlett, The symmetric eigenvalue problem, no. 20 in *Classics in Applied Mathematics*, SIAM, 1998.
- [15] A. Greenbaum, Behavior of slightly perturbed lanczos and conjugate-gradient recurrences, *Linear Algebra and its Applications* 113 (0) (1989) 7 – 63. doi:10.1016/0024-3795(89)90285-1.  
URL <http://www.sciencedirect.com/science/article/pii/0024379589902851>
- [16] A. Greenbaum, Z. Strakos, Predicting the behavior of finite precision lanczos and conjugate gradient computations, *SIAM Journal on Matrix Analysis and Applications* 13 (1) (1992) 121–137. doi:10.1137/0613011.  
URL <http://link.aip.org/link/?SML/13/121/1>
- [17] G. H. Golub, C. F. VanLoan, *Matrix Computations*, 3rd Edition, The Johns Hopkins University Press, Baltimore, MD, 1996.



- [18] H. D. Simon, Analysis of the symmetric Lanczos algorithm with re-orthogonalization methods, *Linear Algebra and Applications* 61 (1984) 101 – 131.
- [19] H. D. Simon, The Lanczos algorithm with partial reorthogonalization, *Mathematics of Computation* 42 (165) (1984) 115 – 142.
- [20] P. Fernandes, B. Plateau, W. J. Stewart, Efficient descriptor-vector multiplications in stochastic automata networks, *J. ACM* 45 (3) (1998) 381–414. doi:doi.acm.org/10.1145/278298.278303.
- [21] J. Shadid, A. Salinger, R. Pawlowski, P. Lin, G. Hennigan, R. Tuminaro, R. Lehoucq, Large-scale stabilized fe computational analysis of nonlinear steady-state transport/reaction systems, *Comput. Methods Appl. Mech. Engrg.* 195 (2006) 1846–1871.
- [22] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 1996.
- [23] R. P. Pawlowski, E. T. Phipps, A. G. Salinger, S. J. Owen, C. Siefert, M. L. Staten, Applying template-based generic programming to the simulation and analysis of partial differential equations, Submitted to *Journal of Scientific Programming*.
- [24] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, K. Stanley, An overview of the Trilinos package, *ACM Trans. Math. Softw.* 31 (3), <http://trilinos.sandia.gov/>.